

Universal Windows Platform:

Design an enterprise architecture for universal app

Aprile Salvatore

Beque.it

.NET Present and Future



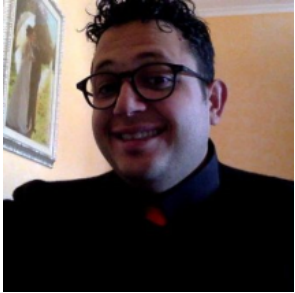
Grazie a



Communities



Chi sono

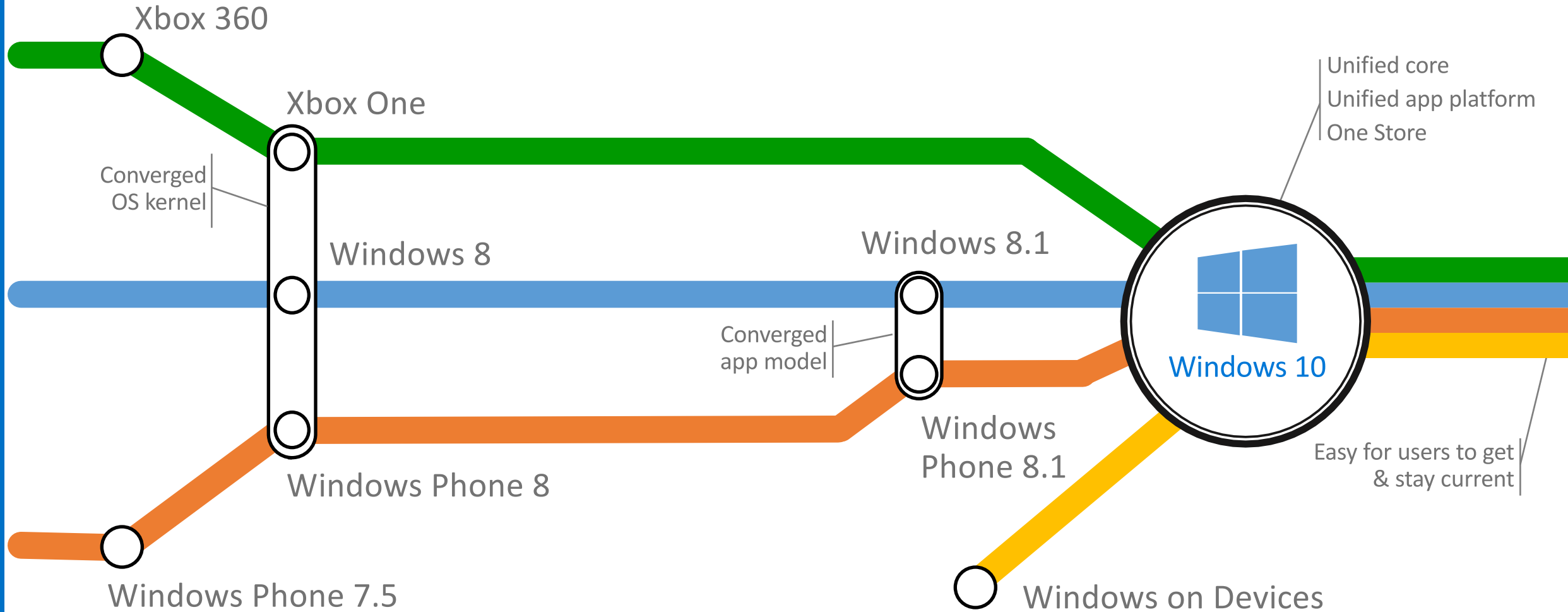


- MCPD - Designing and Developing Web Applications
- MCPD - Designing and Developing Windows Azure Applications

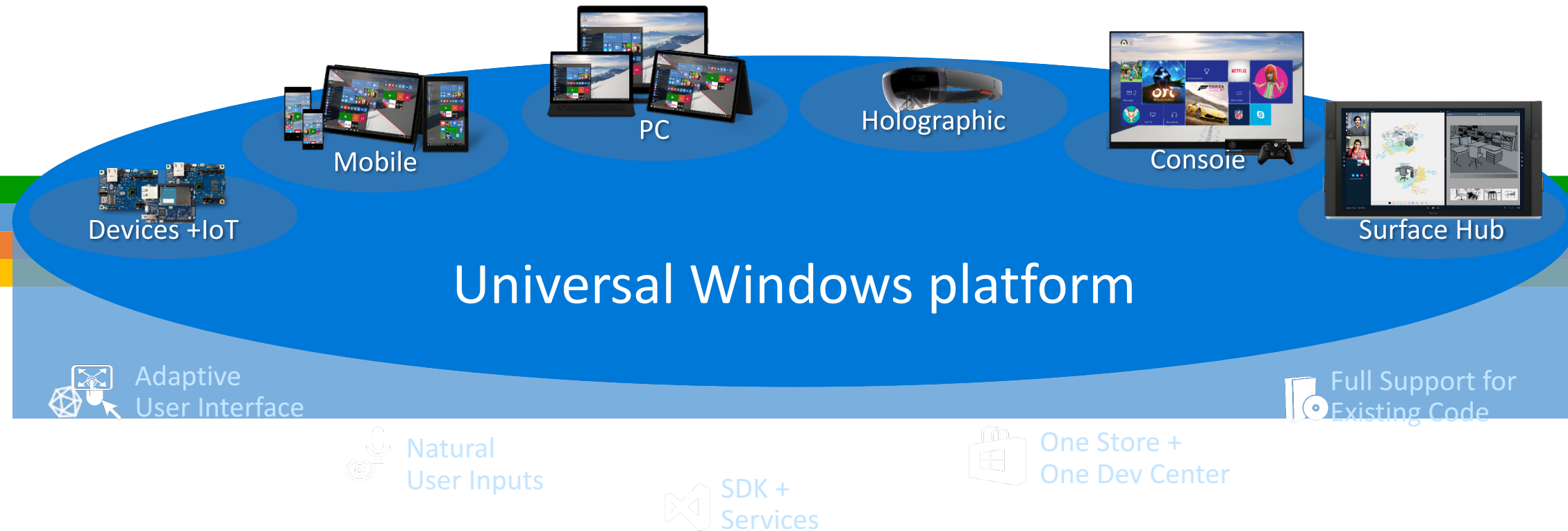
- MCTS ASP.NET, ADO.NET e WCF
- Technical Architect & Mobile Tecnical Solution for Windows and Android
- Contatti:
 - Email: aprile.salvatore@gmail.com
 - Sito: <http://www.beque.it> – <http://trakky-app.com>
 - Skype: aprile.salvatore81
 - Linkedin: it.linkedin.com/in/aprilesalvatore
 - Twitter: twitter.com/SalvoAprile



Un unico sistema operativo...



...e un'unica piattaforma di sviluppo



Universal Windows Platform

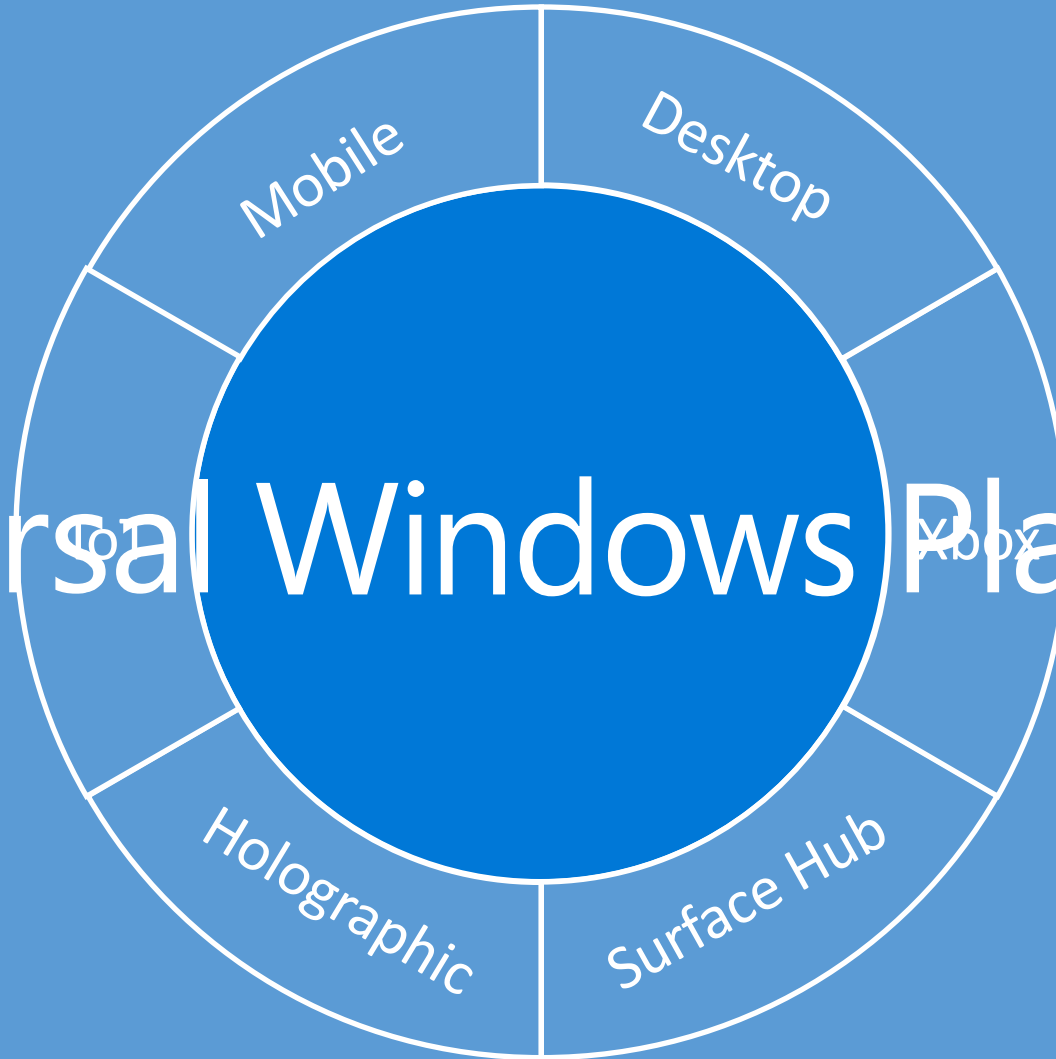
- E' una collezione di contratti ed estensioni, basata sul Windows Runtime
- Quando si crea un'applicazione si sceglie come riferimento una versione della UWP, non del sistema operativo

```
<Dependencies>  
  <TargetDeviceFamily Name="Windows.Universal"  
MinVersion="10.0.10069.0" MaxVersionTested="10.0.10069.0" />  
</Dependencies>
```

Un pacchetto per tutti i dispositivi

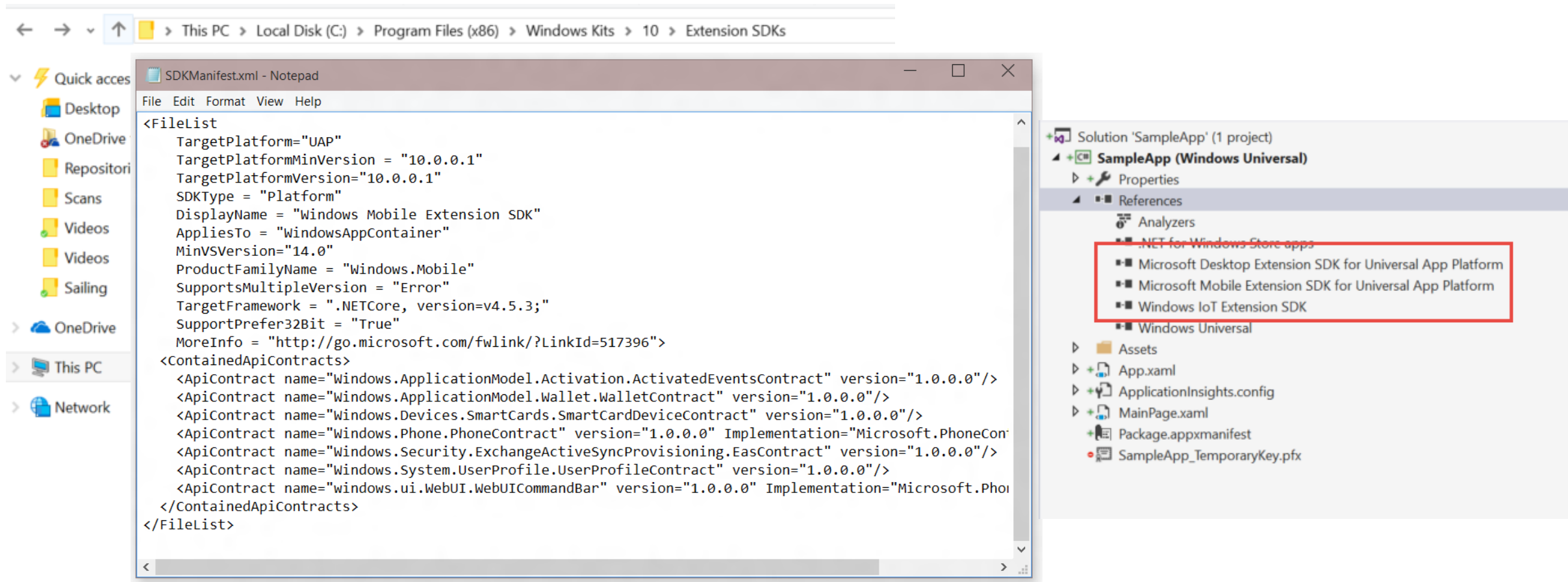
- Scompaiono il progetto condiviso e i progetti specifici per ogni piattaforma
- Niente più compilazione condizionale
- Un unico Store per tutte le piattaforme
- La Universal Windows Platform è disponibile su ogni dispositivo basato su Windows 10

Universal Windows Platform



Platform extension

Librerie che aggiungono funzionalità specifiche di una piattaforma alla Universal Windows Platform



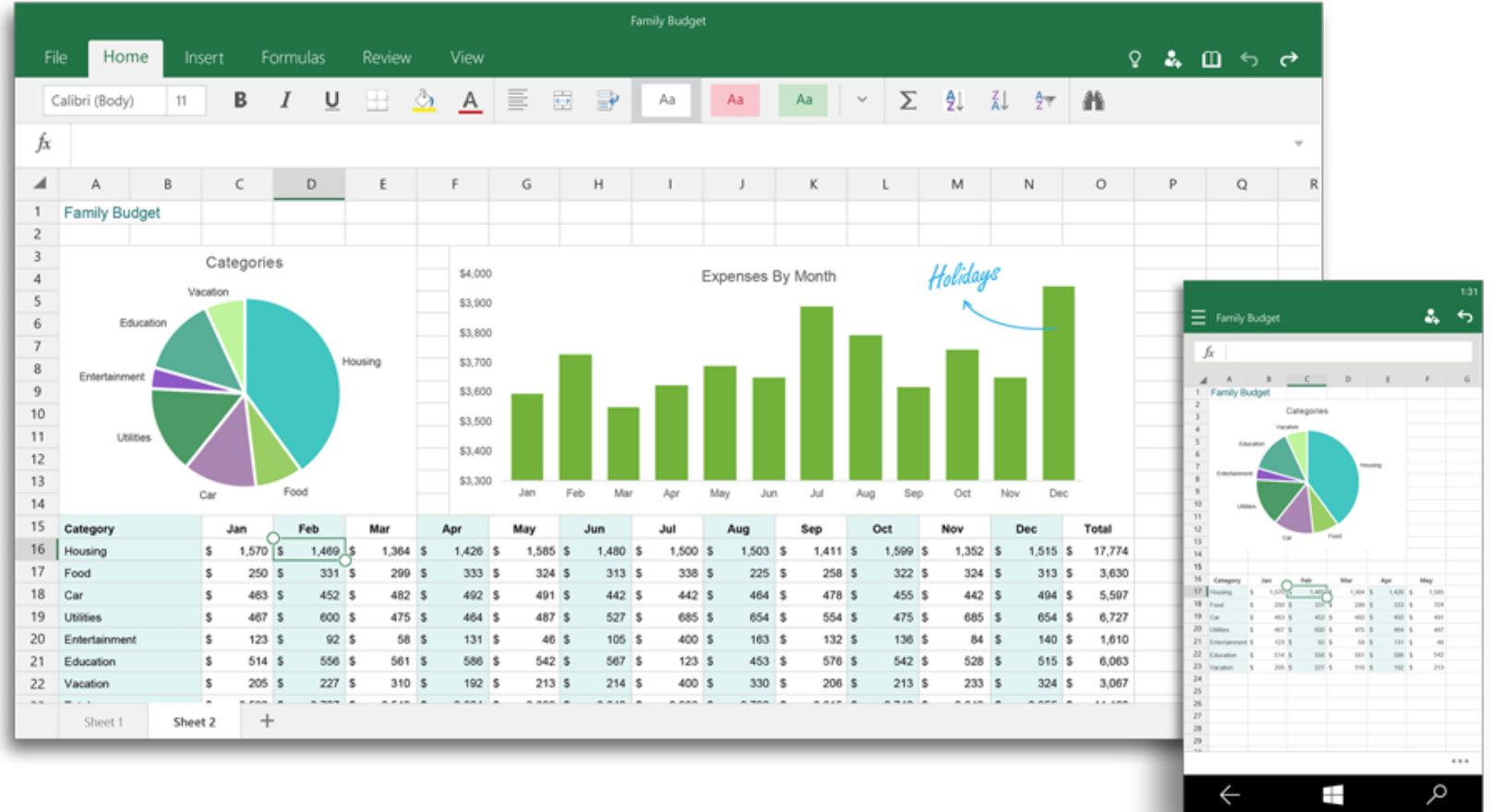
Ottimizzare l'esperienza utente

- Un solo binario significa che l'applicazione deve adattarsi ai diversi dispositivi e dimensioni dello schermo
- Nuovi controlli e tool per creare «responsive app»
- Nuova opzione per collegare la stessa classe di code behind a diversi file XAML

App, app e poi app



FLASHLIGHT XT



e Allora ?

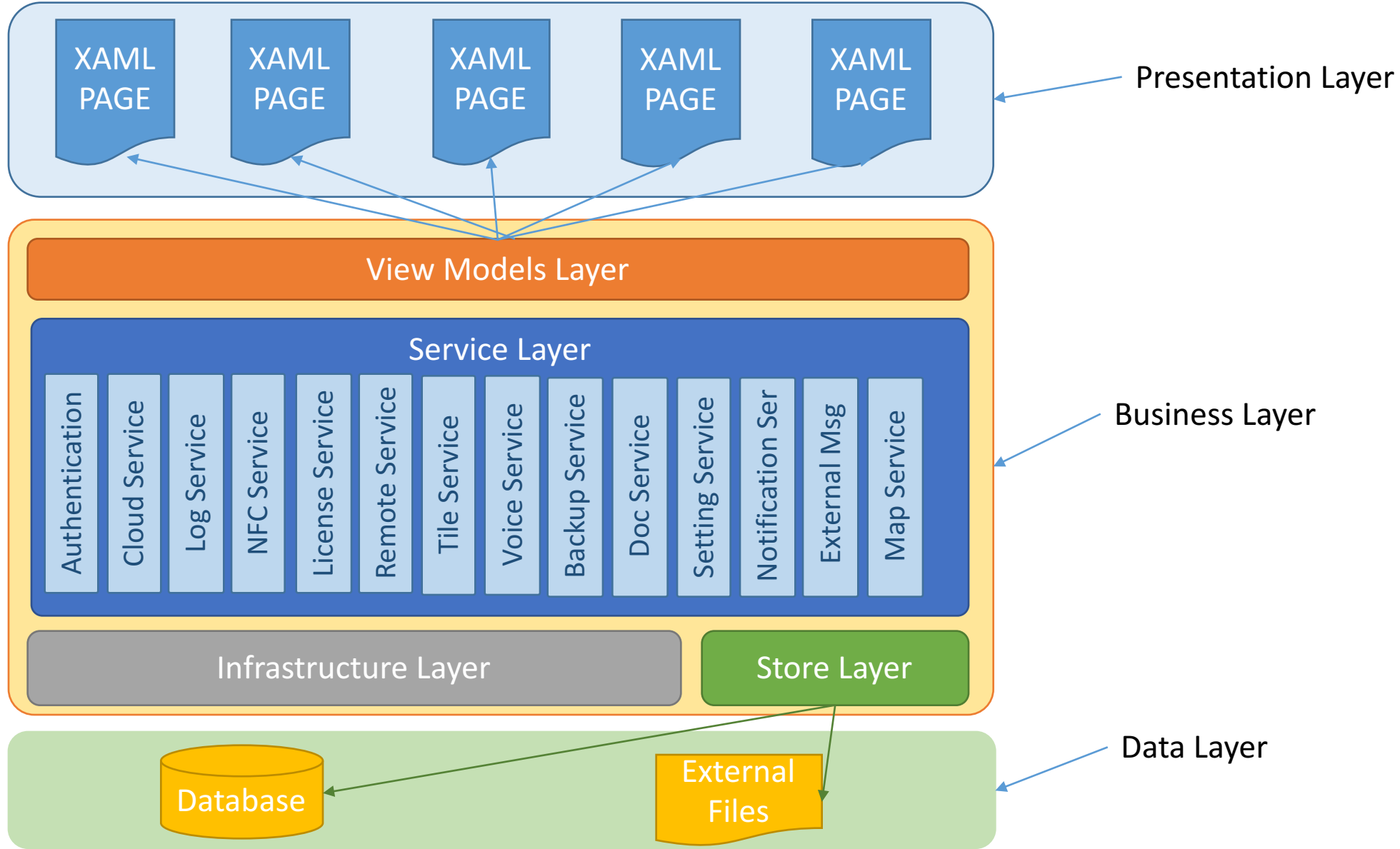
- Perchè allora non portare i concetti architetturali applicati allo sviluppo di enterprise client anche per lo sviluppo di app ?
- Cosa voglio dire ?
 - Separazione dei Layer
 - Applicazione di Pattern
 - Information Hiding
 - Application service oriented
 - Injection
 - Event dispatcher
 - ...

La nostra app – DotNetSide Journal

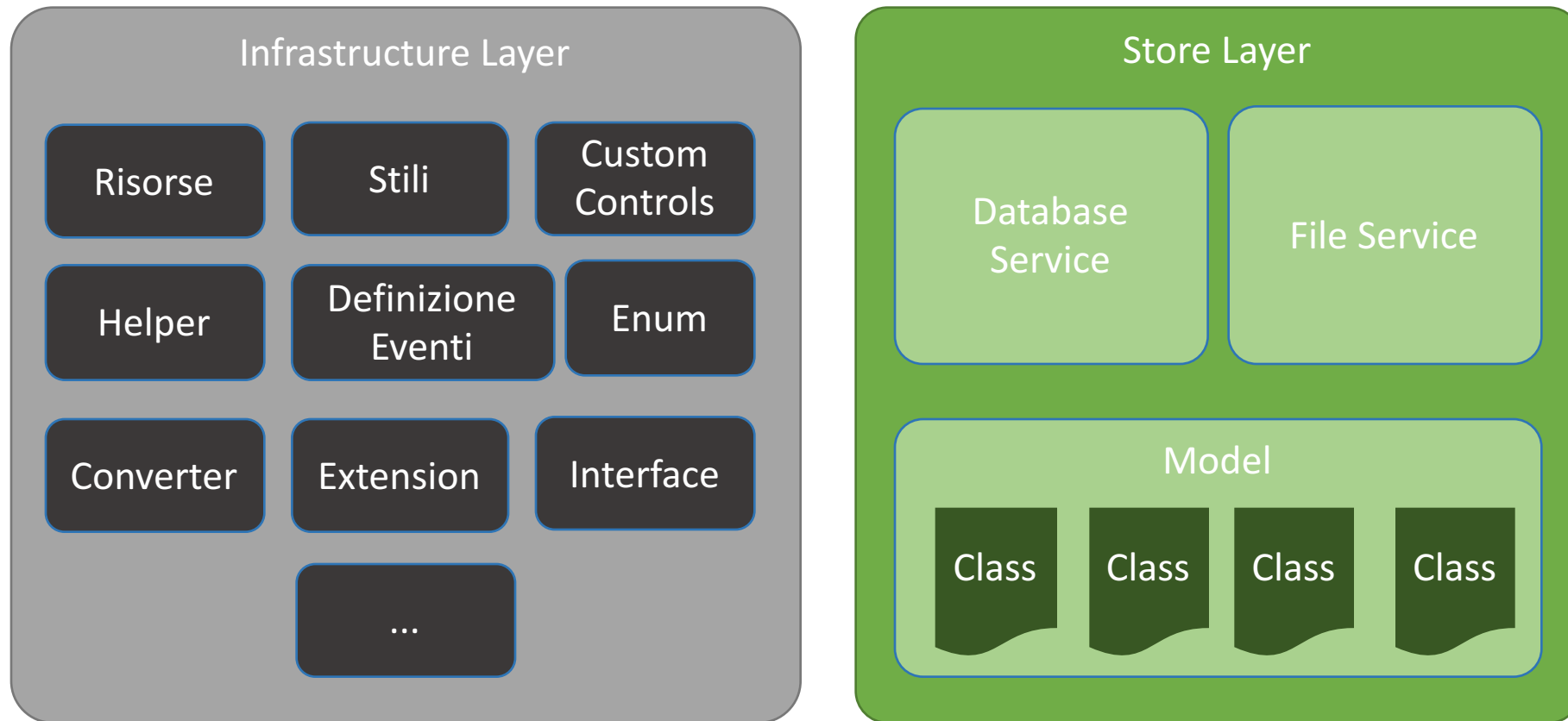
- Immaginiamo di dover sviluppare un app che debba interagire con alcuni servizi dove scaricare la lista degli eventi di dotnetside, con la possibilità di vedere un dettaglio dell'evento, consultare l'agenda e indicare quali sono gli eventi preferiti



DotNetSide Journal – Architettura 1



DotNetSide Journal – Architettura 2



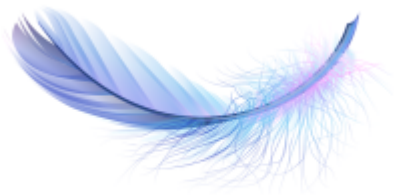
Concetti

- I Concetti che affronteremo sviluppando una architettura simile possono essere raggruppati in :
 - Dependency Injection e IOC
 - Advanced Data Binding
 - View Model Pairing
 - Event Dispatcher
 - Navigation Mechanism

I framework

- Per supportare e implementare tale infrastruttura possiamo aiutarci usando alcuni framework disponibili che saranno oggetto di indagine e comparazione e sono :

Caliburn.Micro Xaml made easy



MVVM Light Toolkit



Prism



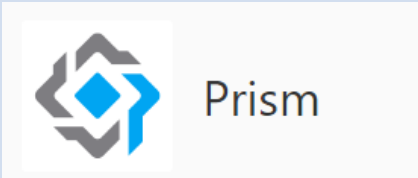



MvvmCross

MvvmCross: The .NET MVVM framework for cross-platform solutions.





<http://mvvmcross.com>

I framework – dove trovarli

FRAMEWORK	NUGET
 <p>MVVM Light Toolkit</p>	https://www.nuget.org/packages/MvvmLight/
 <p>Caliburn.Micro Xaml made easy</p>	https://www.nuget.org/packages/Caliburn.Micro/
 <p>Prism</p>	https://www.nuget.org/packages/Prism.Windows/
 <p>MvvmCross MvvmCross: The .NET MVVM framework for cross-platform solutions. http://mvvmcross.com</p>	https://www.nuget.org/packages/MvvmCross/

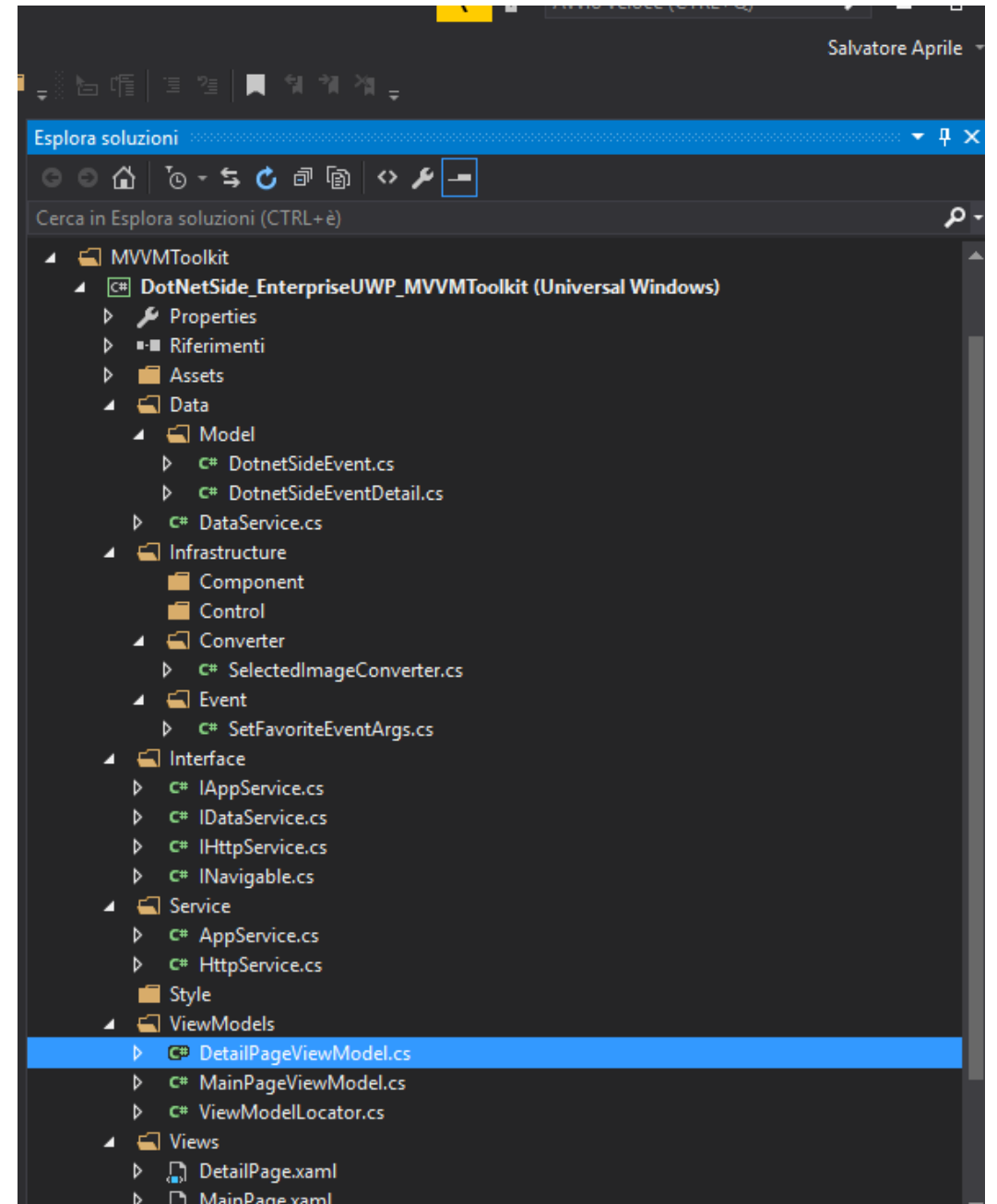
Valutiamo i framework

- I framework saranno valutati sulle seguenti voci con un punteggio da 1 a 5. Il vincitore sarà quello che ha totalizzato un punteggio maggiore

FRAMEWORK	Dependency Injection	Advanced Data Binding	View Model Pairing	Event Dispatcher	Navigation	Compatibilità
 MvvmLight Toolkit						
 Caliburn.Micro Xaml made easy						
 Prism						
 MvvmCross MvvmCross: The .NET MVVM framework for cross-platform solutions. http://mvvmcross.com						

La solution

- Sarà composta da 4 progetti che avranno il 95 % del codice in comune e che rispecchieranno i 4 framework analizzati



DEMO

Configurazione Iniziale e Dependency Injection - Caliburn

- La configurazione iniziale non è immediata
- La class App deve ereditare da CaliburnApplication e non mantiene l'originale
- Bisogna effettuare l'override di almeno 6 metodi
- + La configurazione del container IOC è molto semplice ed è effettuata nel metodo Configure della classe App
- + Container IOC molto flessibile

```
/// Fornisci un comportamento specifico dell'applicazione in supplemento alla c
/// </summary>
public sealed partial class App
{
    private WinRTContainer _container;

    public App()
    {
        InitializeComponent();
    }

    protected override void Configure()
    {
        _container = new WinRTContainer();
        _container.RegisterWinRTServices();

        _container.PerRequest<MainPageViewModel>();
        _container.PerRequest<DetailPageViewModel>();

        _container.Singleton<IDataService, DataService>();
        _container.Singleton<IAppService, AppService>();
        _container.Singleton<IHttpService, HttpService>();
    }

    protected override object GetInstance(Type service, string key)
    {
        var instance = _container.GetInstance(service, key);
        if (instance != null)
            return instance;
        throw new Exception("Could not locate any instances.");
    }

    protected override IEnumerable<object> GetAllInstances(Type service)
    {

```

Configurazione Iniziale e Dependency Injection – MVVM Light toolkit

- + La configurazione iniziale è veramente immediata
- + Bisogna solo creare una classe chiamata ViewModelLocator e metterla come risorsa dell'app
- + Nessuna modifica della classe App
- + La configurazione del container IOC è molto semplice ed è effettuata nel costruttore del ViewModelLocator
- + Container IOC di buona qualità

```
Application
{
    x:Class="DotNetSide_EnterpriseUWP_MVVMToolkit.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:DotNetSide_EnterpriseUWP_MVVMToolkit;assembly=DotNetSide_EnterpriseUWP_MVVMToolkit"
    x:Name="App"
    x:DataType="local:Application"
    local:ViewModelLocator x:Name="ViewModelLocator"
}

public class ViewModelLocator
{
    public const string MainPageKey = "MainPage";
    public const string DetailPageKey = "DetailPage";

    /// <summary>
    /// Initializes a new instance of the ViewModelLocator class.
    /// </summary>
    public ViewModelLocator()
    {
        ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);
        if (ViewModelBase.IsInDesignModeStatic)
        {
            // Create design time view services and models
        }
        else
        {
            // Create run time view services and models
        }

        var nav = new NavigationService();
        nav.Configure(MainPageKey, typeof(MainPage));
        nav.Configure(DetailPageKey, typeof(DetailPage));

        //Register your services used here
        SimpleIoc.Default.Register<INavigationService>(() => nav);
        SimpleIoc.Default.Register<IDataService, DataService>();
        SimpleIoc.Default.Register<IAppService, AppService>();
        SimpleIoc.Default.Register<IHttpService, HttpService>();

        SimpleIoc.Default.Register<MainPageViewModel>();
        SimpleIoc.Default.Register<DetailPageViewModel>();
    }
}
```

Configurazione Iniziale e Dependency Injection – MVVM Cross

- La configurazione iniziale difficile anche a causa della documentazione molto confusionaria e non aggiornata
- Bisogna creare 2 file (setup e app) e modificare la classe App.cs di default
- Questo a causa del fatto che questo framework è stato studiato e pensato per Xamarin e quindi con la gestione PCL e Platform dependent
- Configurazione IOC non lineare e abbastanza confusionaria anche se ricca di opzioni

```
public class CrossApp : MvxApplication
{
    public override void Initialize()
    {
        CreatableTypes()
            .EndingWith("Service")
            .AsInterfaces()
            .RegisterAsLazySingleton();

        Mvx.LazyConstructAndRegisterSingleton<IDataService, DataService>();
        Mvx.LazyConstructAndRegisterSingleton<IAppService, AppService>();
        Mvx.LazyConstructAndRegisterSingleton<IHttpService, HttpService>();

        RegisterAppStart<MainViewModel>();
    }
}
```

```
public class Setup : MvxWindowsSetup
{
    public Setup(Frame rootFrame) : base(rootFrame)
    {
    }

    protected override IMvxApplication CreateApp()
    {
        return new CrossApp();
    }
}
```


Configurazione Iniziale e Dependency Injection - Prism

- La configurazione iniziale non è immediata
 - La class App deve ereditare da PrismUnityApplication e non mantiene l'originale
 - Bisogna effettuare l'override di almeno 2 metodi
- + La configurazione del container IOC è molto semplice ed è effettuata nel metodo Configure della classe App
- + Container IOC molto flessibile

```
sealed partial class App : PrismUnityApplication
{
    public App()
    {
        this.InitializeComponent();
    }

    protected override Task OnLaunchApplicationAsync(LaunchActivatedEventArgs args)
    {
        NavigationService.Navigate("Main", null);
        Window.Current.Activate();
        return Task.FromResult(true);
    }

    //protected override UIElement CreateShell(Frame rootFrame)
    //{
    //    var shell = Container.Resolve<MainPage>();

    //    return shell;
    //}




    protected override Task OnInitializeAsync(IActivatedEventArgs args)
    {
        var _eventAggregator = new EventAggregator();

        Container.RegisterInstance<INavigationService>(NavigationService);
        Container.RegisterInstance<IEventAggregator>(EventAggregator);

        Container.RegisterType<IMainPageViewModel, MainPageViewModel>();
        Container.RegisterType<IDetailPageViewModel, DetailPageViewModel>();

        Container.RegisterType<IHttpService, HttpService>(new ContainerControlledLifetimeManager());
        Container.RegisterType<IAppService, AppService>(new ContainerControlledLifetimeManager());
        Container.RegisterType<IDataService, DataService>(new ContainerControlledLifetimeManager());
    }
}
```

Configurazione Iniziale e Dependency Injection – I voti

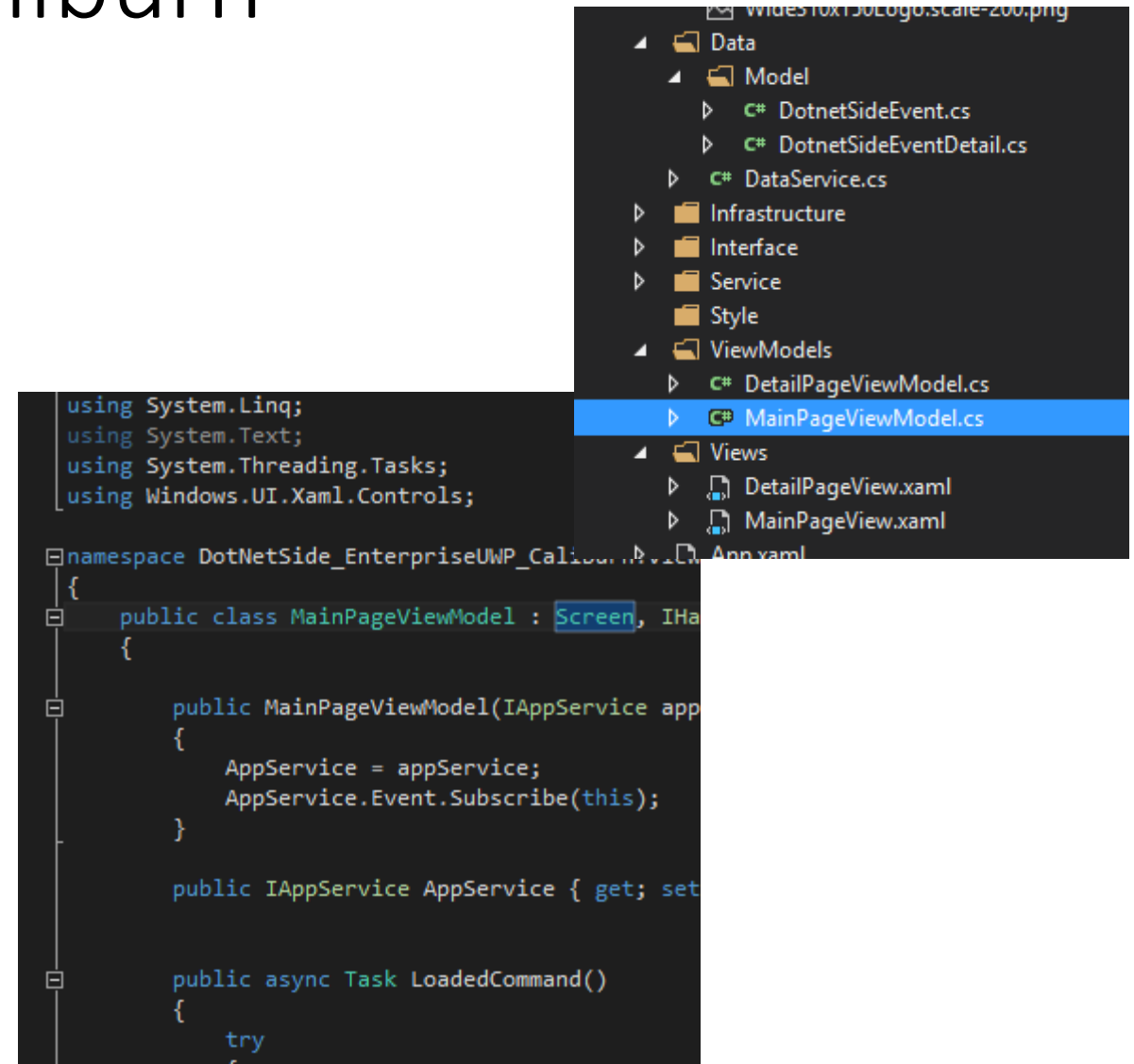
FRAMEWORK	Dependency Injection	View Model Pairing	Advanced Data Binding	Event Dispatcher	Navigation	Compatibilità
	5					
Caliburn.Micro Xaml made easy	3,5					
 Prism	4					
 MvvmCross MvvmCross: The .NET MVVM framework for cross-platform solutions. http://mvvmcross.com	2					

View Model Pairing- Caliburn

+ Il Pairing del view model con la view è del tutto automatico basato su una naming convention e senza scrivere una linea di codice

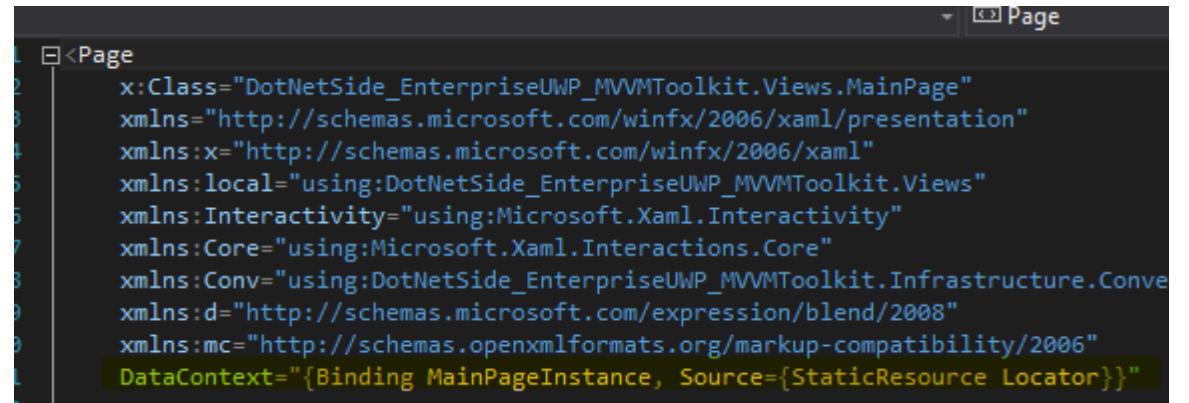
Le view devono terminare con la parola View e essere contenute in una cartella Views

I view model devono avere lo stesso nome della view + Model alla fine e essere contenuti nella cartella ViewModels



View Model Pairing– MVVM Light toolkit

- Non c'è nessuna facility, il tutto avviene attraverso il binding della proprietà DataContext della pagina con un'istanza di ViewModel esposta dal View Locator



```
<Page
    x:Class="DotNetSide_EnterpriseUWP_MVVMToolkit.Views.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:DotNetSide_EnterpriseUWP_MVVMToolkit.Views"
    xmlns:Interactivity="using:Microsoft.Xaml.Interactivity"
    xmlns:Core="using:Microsoft.Xaml.Interactions.Core"
    xmlns:Conv="using:DotNetSide_EnterpriseUWP_MVVMToolkit.Infrastructure.Conve"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    DataContext="{Binding MainPageInstance, Source={StaticResource Locator}}">
```

View Model Pairing– MVVM Cross

- + Il Pairing del view model con la view è del tutto automatico basato su una naming convention e senza scrivere una linea di codice
- + Le view devono terminare con la parola View e essere contenute in una cartella Views
- + I view model devono avere lo stesso nome della view + Model alla fine e essere contenuti nella cartella ViewModels
- Rispetto a caliburn, le view non oggetti standard ma derivano da oggetti proprietari del framework




```
/// <summary>  
/// Pagina vuota che può essere usata autonomamente oppure per 1  
/// </summary>  
public sealed partial class MainView : MvxWindowsPage  
{  
    public MainView()  
    {  
        this.InitializeComponent();  
    }  
}
```

View Model Pairing- Prism

- + Il Pairing del view model con la view è automatico basato su una naming convention
- + Le view devono terminare con la parola View e essere contenute in una cartella Views
- + I view model devono avere lo stesso nome della view + Model alla fine e essere contenuti nella cartella ViewModels
- Bisogna abilitare il l'AutoWire sulla pagina

```
<Page
    x:Class="DotNetSide_EnterpriseUWP_Prism.Views.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:DotNetSide_EnterpriseUWP_Prism.Views"
    xmlns:Interactivity="using:Microsoft.Xaml.Interactivity"
    xmlns:Core="using:Microsoft.Xaml.Interactions.Core"
    xmlns:Conv="using:DotNetSide_EnterpriseUWP_Prism.Infrastructure.Converters"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mvvm="using:Prism.Windows.Mvvm"
    mvvm:ViewModelLocator.AutoWireViewModel="True"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

View Model Pairing– I voti

FRAMEWORK	Dependency Injection	View Model Pairing	Advanced Data Binding	Event Dispatcher	Navigation	Compatibilità
 MVVM Light Toolkit	5	1,5				
Caliburn.Micro Xaml made easy	3,5	5				
 Prism	4	4,5				
 MvvmCross MvvmCross: The .NET MVVM framework for cross-platform solutions. http://mvvmcross.com	2	4,5				

Advance Data Binding - Caliburn

+ Il binding di una proprietà dello xaml può avvenire attraverso naming convention

+ Possiede un meccanismo di binding degli eventi unico nel suo genere. Attraverso il parse di una string di una attached property è in grado di effettuare l'event binding direttamente con un metodo del view model senza l'utilizzo di command. Strumento potentissimo che fa risparmiare tantissimo codice

Databinding

This is automatically binding dependency properties on controls to properties on the ViewModel.

Convention

```
<TextBox x:Name="FirstName" />
```

Will cause the "Text" property of the TextBox to be bound to the "FirstName" property on the ViewModel.

Ex

```
<Button cal:Message.Attach="[Event MouseEnter] = [Action Save($this)]">
```

<1

- » **SeventArgs** – Passes the EventArgs or input parameter to your Action. Note: This will be null for guard methods since the trigger hasn't actually occurred.
- » **SdataContext** – Passes the DataContext of the element that the ActionMessage is attached to. This is very useful in Master/Detail scenarios where the ActionMessage may bubble to a parent VM but needs to carry with it the child instance to be acted upon.
- » **Ssource** – The actual FrameworkElement that triggered the ActionMessage to be sent.
- » **Sview** – The view (usually a UserControl or Window) that is bound to the ViewModel.
- » **SexecutionContext** – The action's execution context, which contains all the above information and more. This is useful in advanced scenarios.
- » **Sthis** – The actual UI element to which the action is attached. In this case, the element itself won't be passed as a parameter, but rather its default property.

Advance Data Binding– MVVM Light toolkit

- Si appoggia al meccanismo di uwp per il dispatcher di un evento nel corrispettivo command bindato
- + Mette a disposizione una implementazione dell' interfaccia ICommand chiamata RelayCommand che può essere bindata ad un evento

```
<Interactivity:Interaction.Behaviors>
  <Core:EventTriggerBehavior EventName="Loaded">
    <Core:InvokeCommandAction Command="{Binding LoadedCommand}" />
  </Core:EventTriggerBehavior>
</Interactivity:Interaction.Behaviors>
```

```
private RelayCommand<object> _loadedCommand;
public RelayCommand<object> LoadedCommand
{
    get
    {
        return _loadedCommand ?? (_loadedCommand = new RelayCommand<object>(
            async x =>
            {
                try
                {
                    Title = "DotNetSide Journal";

                    SubTitle = "MVVM Light Toolkit edition";

                    AppService.Data.CreateDatabase();

                    await LoadEvents();
                }
                catch (Exception ex)
                {
                    new Windows.UI.Popups.MessageDialog("Ops !! come sempre in una demo");
                }
            }
        ));
    }
}
```

Advance Data Binding– MVVM Cross

- Si appoggia al meccanismo di uwp per il dispatcher di un evento nel corrispettivo command bindato
- + Mette a disposizione una implementazione dell' interfaccia ICommand chiamata MvxCommand che può essere bindata ad un evento
- Per altre piattaforme (Android) mette a disposizione una implementazione simile al Message.Attach di Caliburn attraverso un plugin esterno, ma non è disponibile per UWP

```
private MvxCommand<object> _loadedCommand;  
public MvxCommand<object> LoadedCommand  
{  
    get  
    {  
        return _loadedCommand ?? (_loadedCommand = new MvxCommand<object>(  
            async x =>  
            {  
                try  
                {  
                    SubTitle = "MVVMCross edition";  
  
                    if (_event != null)  
                    {  
                        EventDetail = await AppService.Http.GetEventDetail(_event.Id, _event.LinkDetail);  
                    }  
                }  
                catch (Exception ex)  
                {  
                    new Windows.UI.Popups.MessageDialog("Ops !! come sempre in una demo ci deve essere se  
                }  
            }  
        ));  
    }  
}
```

```
<Button  
    android:layout_width='fill_parent'  
    android:layout_height='wrap_content'  
    android:text='Save'  
    local:MvxBind='Click Save' />
```

Advance Data Binding - Prism

- Si appoggia al meccanismo di uwp per il dispatcher di un evento nel corrispettivo command bindato
- + Mette a disposizione una implementazione dell' interfaccia ICommand chiamata DelegateCommand che può essere bindata ad un evento

```
<Page.Resources>
    <Conv:SelectedImageConverter x:Key="selectedImageConverter"></Conv:SelectedImageConverter>
</Page.Resources>

<Interactivity:Interaction.Behaviors>
    <Core:EventTriggerBehavior EventName="Loaded">
        <Core:InvokeCommandAction Command="{Binding LoadedCommand}" />
    </Core:EventTriggerBehavior>
</Interactivity:Interaction.Behaviors>
```




```
private DelegateCommand<object> _loadedCommand;
public DelegateCommand<object> LoadedCommand
{
    get
    {
        return _loadedCommand ?? (_loadedCommand = new DelegateCommand<object>
            async x =>
            {
                try
                {
                    Title = "DotNetSide Journal";

                    SubTitle = "Prism edition";

                    AppService.Data.CreateDatabase();

                    await LoadEvents();
                }
                catch (Exception ex)
                {
                    new Windows.UI.Popups.MessageDialog("Ops !! come mai")
                        .ShowAsync();
                }
            }
            ));
    }
}
```

Advance Data Binding – I voti

FRAMEWORK	Dependency Injection	View Model Pairing	Advanced Data Binding	Event Dispatcher	Navigation	Compatibilità
 MVVM Light Toolkit	5	1,5	3,5			
Caliburn.Micro Xaml made easy	3,5	5	5			
 Prism	4	4,5	3,5			
 MvvmCross MvvmCross: The .NET MVVM framework for cross-platform solutions. http://mvvmcross.com	2	4,5	3			

Event Dispatcher - Caliburn

- + Supporta la possibilità di effettuare un dispatcher di un evento con un meccanismo di publish / subscribe
- Rispetto ai suoi concorrenti occorre scrivere un po di codice in più per far funzionare il tutto

La classe che si sottoscrive deve implementare una interfaccia dell'eventargs usato nel messaggio che sarà poi il metodo dove andrà a finire il messaggio

```
public class MainPageViewModel : Screen, IHandle<SetFavoriteEventArgs>
{
    public MainPageViewModel(IAppService appService)
    {
        AppService = appService;
        AppService.Event.Subscribe(this);
    }
}
```

```
public void Handle(SetFavoriteEventArgs message)
{
    if (message != null)
    {
        AppService.Data.SetIsFavorite(message.Id, message.IsFavorite);

        var e1 = Events.FirstOrDefault(x => x.Id == message.Id);

        if (e1 != null)
        {
            e1.IsFavorite = message.IsFavorite;
        }
    }
}
```

```
if (EventDetail != null)
{
    AppService.Event.PublishOnUIThread(new SetFavoriteEventArgs() { Id = EventDetail.Id, IsFavorite = Convert.ToBoolean(e) });
}
```

Event Dispatcher– MVVM Light toolkit

+ Supporta la possibilità di effettuare un dispatcher di un messaggio in maniera molto semplice e completa

+ Bisogna scrivere soltanto la classe che sarà l'eventArgs, il dispatcher si basa su questa

+ Nessuna interfaccia da implementare

```
public class MainPageViewModel : ViewModelBase
{
    public MainPageViewModel(IAppService appService)
    {
        AppService = appService;
        Messenger.Default.Register<SetFavoriteEventArgs>(this, SetFavorite);
    }

    public IAppService AppService { get; set; }
}
```

```
private void SetFavorite(SetFavoriteEventArgs e)
{
    if(e != null)
    {
        AppService.Data.SetIsFavorite(e.Id, e.IsFavorite);

        var el = Events.FirstOrDefault(x => x.Id == e.Id);

        if(el != null)
        {
            el.IsFavorite = e.IsFavorite;
        }
    }
}
```

```
{
    if (EventDetail != null)
    {
        Messenger.Default.Send(new SetFavoriteEventArgs() { Id = EventDetail.Id, IsFavorite = Convert.ToBoolean(x) });
    }
}
```

Event Dispatcher– MVVM Cross

- + Supporta la possibilità di effettuare un dispatcher di un messaggio in maniera molto semplice e completa
- + Basato su un plugin scaricabile (Messenger)
- + Bisogna scrivere la classe che sarà l'eventArgs, il dispatcher si basa su questa
- Necessita la creazione di una classe bootstrap per caricare il plugin
- Il dispatcher non sempre avviene (Bug ?)
- + Nessuna interfaccia da implementare

```
public class MessengerPluginBootstrap
    : MvxPluginBootstrapAction<MvvmCross.Plugins.Messenger.PluginLoader>
{
    private readonly MvxSubscriptionToken _token;

    public MainViewModel(IAppService appService)
    {
        AppService = appService;
        _token = AppService.Messenger.SubscribeOnMainThread<SetFavoriteEventArgs>(SetFavorite);
    }

    public IAppService AppService { get; set; }

    private void SetFavorite(SetFavoriteEventArgs e)
    {
        if (e != null)
        {
            AppService.Data.SetIsFavorite(e.Id, e.IsFavorite);

            var el = Events.FirstOrDefault(x => x.Id == e.Id);

            if (el != null)
            {
                el.IsFavorite = e.IsFavorite;
            }
        }
    }
}
```

```
if (EventDetail != null)
{
    AppService.Messenger.Publish(new SetFavoriteEventArgs(this, EventDetail.Id, Convert.ToBoolean(x), Guid.NewGuid()));
}
```

Event Dispatcher - Prism

- + Supporta la possibilità di effettuare un dispatcher di un evento
- Bisogna creare una classe che derivi da PubSubEvent e che rappresenterà il nostro evento
- Bisogna creare anche la classe che sarà l'eventArgs usata dalla classe precedente

```
if (EventDetail != null)
{
    AppService.Event.GetEvent<SetFavoriteEvent>().Publish(new SetFavoriteEventArgs() { Id = E
}
```

```
public MainPageViewModel(IAppService appService)
{
    AppService = appService;
    AppService.Event.GetEvent<SetFavoriteEvent>().Subscribe(SetFavorite);
}
```





```
using System.Threading.Tasks;

namespace DotNetSide_EnterpriseUWP_Prism.Infrastructure.Event
{
    public class SetFavoriteEvent: PubSubEvent<SetFavoriteEventArgs>
    {
    }
}

EventArgs.cs -> X
DotNetSide_EnterpriseUWP_Prism
using Prism.Events;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DotNetSide_EnterpriseUWP_Prism.Infrastructure.Event
{
    public class SetFavoriteEventArgs
    {
        public string Id { get; set; }
        public bool IsFavorite { get; set; }
    }
}
```


Event Dispatcher– I voti

FRAMEWORK	Dependency Injection	View Model Pairing	Advanced Data Binding	Event Dispatcher	Navigation	Compatibilità
 <p>MVVM Light Toolkit</p>	5	1,5	3,5	5		
 <p>Caliburn.Micro Xaml made easy</p>	3,5	5	5	4		
 <p>Prism</p>	4	4,5	3,5	4		
 <p>MvvmCross MvvmCross: The .NET MVVM framework for cross-platform solutions. http://mvvmcross.com</p>	2	4,5	3	4		

Navigation Mechanism - Caliburn

+ Supporta la possibilità navigare per viewmodel oltre che per view

```
{  
    AppService.Navigation.NavigateToViewModel<DetailPageViewModel>(e.ClickedItem);  
}
```

+ Implementa un meccanismo automatico di binding nel view model ricevente del parametro passato nel metodo di navigazione. É sufficiente avere una proprietà dello stesso tipo del parametro passato e che abbia come nome Parameter. Non necessita di nessun altro codice aggiuntivo

```
public class DetailPageViewModel : Screen  
{  
    public DetailPageViewModel(IAppService appService)  
    {  
        AppService = appService;  
    }  
    public IAppService AppService { get; set; }  
    public DotnetSideEvent Parameter { get; set; }  
    |  
    public async Task LoadedCommand()  
    {  
        ...  
    }  
}
```

Navigation Mechanism– MVVM Light toolkit

- Non Supporta la possibilità navigare per viewmodel
- La navigazione avviene per «nome della vista»
- Non supporta il binding del parametro sul view model. **BISOGNA SPORCARE IL CODE BEHIND DELLA VISTA PER IL PASSAGGIO DEL PARAMETRO!!!**

```
/// </summary>
public sealed partial class DetailPage : Page
{
    public DetailPage()
    {
        this.InitializeComponent();
    }

    protected override void OnNavigatedTo(NavigationEventArgs e)
    {
        base.OnNavigatedTo(e);

        var navigableViewModel = this.DataContext as INavigable;
        if (navigableViewModel != null)
            navigableViewModel.Activate(e.Parameter);
    }
}
```

```
try
{
    AppService.Navigation.NavigateTo(ViewModelLocator.DetailPageKey, x.ClickedItem);
}
```

Navigation Mechanism– MVVM Cross

- + Supporta la possibilità navigare per viewmodel
- Non Implementa un meccanismo automatico di binding nel view model ricevente con il parametro passato nel metodo di navigazione, ma necessita di implementare un metodo Init nel view model del ricevente

```
try  
{  
    ShowViewModel<DetailViewModel>(x.ClickedItem);  
}  
catch (Exception ex)
```

```
public void Init(DotnetSideEvent parameter)  
{  
    if (parameter != null)  
    {  
        _event = parameter;  
        Title = _event.Title;  
    }  
}
```

Navigation Mechanism - Prism




- + Supporta la possibilità navigare per viewmodel
- Non implementa un meccanismo automatico di binding nel view model ricevente con il parametro passato nel metodo di navigazione, ma necessita di implementare un override del metodo `OnNavigateTo` (sul viewmodel e non sulla vista) simile al meccanismo implementato da `MVVMCross`

```
try
{
    AppService.Navigation.Navigate("Detail", x.ClickedItem);
}
```

```
public override void OnNavigatedTo(NavigatedToEventArgs e, Dictionary<string, object> viewModelState)
{
    if (e.Parameter != null)
    {
        _event = e.Parameter as DotnetSideEvent;
        Title = _event.Title;
    }

    base.OnNavigatedTo(e, viewModelState);
}
```




Navigation Mechanism– I voti

FRAMEWORK	Dependency Injection	View Model Pairing	Advanced Data Binding	Event Dispatcher	Navigation	Compatibilità
 MVVM Light Toolkit	5	1,5	3,5	5	1	
Caliburn.Micro Xaml made easy	3,5	5	5	4	5	
 Prism	4	4,5	3,5	4	3,5	
 MvvmCross MvvmCross: The .NET MVVM framework for cross-platform solutions. http://mvvmcross.com	2	4,5	3	4	3,5	

Compatibilità

- Caliburn
 - È compatibile con tutto ciò che abbia a che fare con lo Xaml (anche se per alcune piattaforme ci sono alcune limitazioni), in particolare: WPF, Silverlight, WP7, WP8, WP8.1, WinRT, UWP, Xamarin Classic, Xamarin Form
- MVVM Light Toolkit
 - È compatibile con tutto ciò che abbia a che fare con lo Xaml (anche se per alcune piattaforme ci sono alcune limitazioni), in particolare: WPF, Silverlight, WP7, WP8, WP8.1, WinRT, UWP, Xamarin Classic, Xamarin Form
- MVVM Cross
 - Pensata e progettata per supportare tutte le piattaforme
- Prism
 - Ha appena rilasciato la compatibilità per xamarin ma è ancora in fase Preview

Compatibilità– I voti

FRAMEWORK	Dependency Injection	View Model Pairing	Advanced Data Binding	Event Dispatcher	Navigation	Compatibilità
 MVVM Light Toolkit	5	1,5	3,5	5	1	4,5
Caliburn.Micro Xaml made easy	3,5	5	5	4	5	4
 Prism	4	4,5	3,5	4	3,5	3,5
 MvvmCross MvvmCross: The .NET MVVM framework for cross-platform solutions. http://mvvmcross.com	2	4,5	3	4	3,5	5

Caliburn – Pro e Contro

- Pro

- Molte Features uniche nel proprio genere
- Framework maturo
- Buona documentazione
- Buon supporto della community

- Contro

- Rilasci delle versioni abbastanza lenti
- Alcune features non presenti in alcune piattaforme (soprattutto per Xamarin)

MVVM Light Toolkit – Pro e Contro

- Pro

- Impatto sul progetto minimo
- Message dispatcher semplice e molto flessibile
- Molto supporto dalla community e da microsoft stessa

- Contro

- Nessuna facility
- Framework forse troppo «Semplice»
- L'unico che ha necessitato di sporcare il code behind della pagina per il passaggio di parametri in ambito della navigazione

MVVM Cross – Pro e Contro

- Pro
 - Ricchissima documentazione
 - Pensata e sviluppata per applicazioni cross platform
 - Espandibile con Plugin
- Contro
 - Documentazione molto dispersiva e non aggiornata
 - Curva di apprendimento molto alta
 - Gran numero di pacchetti nuget da scaricare
 - In generale molto confusionaria

Prism – Pro e Contro

- Pro

- Official Microsoft Framework
- Framework solido e storico
- Ottima documentazione e esempi

- Contro

- Non ha alcune feature implementati in altri framework
- Tutto piuttosto «standard»
- Per xamarin è ancora in preview

The winner is ...



Caliburn
26,5



Prism
23



MVVM
Cross
22



MVVM
Light
Toolkit
20,5

Grazie, Domande?

- Contatti:
 - **Email:** aprile.salvatore@gmail.com
 - **Skype:** [aprile.salvatore81](https://www.skype.com/people/aprile.salvatore81)
 - **Linkedin:** [it.linkedin.com/in/aprilesalvatore](https://www.linkedin.com/in/aprilesalvatore)
 - **Twitter:** twitter.com/SalvoAprile